**DevLab Automation Guide**

Devlab Automation is developed to automate regression UI functional cases for UDA application.

Automation Tools: Selenium Web driver, TestNG

Framework: Data Driven Framework

Scripting Language: Java

Software requirement:

* JAVA – JDK & JRE --- Version 8 0r above
* Eclipse – Luna or above
* Selenium Jars – Selenium-java 3.141
* POI jars – Version 4
* Extent Report Jars – Version 2.41.2
* Chrome driver – Chrome driver Version 89.0.4389.23

Framework and Tool Configuration:

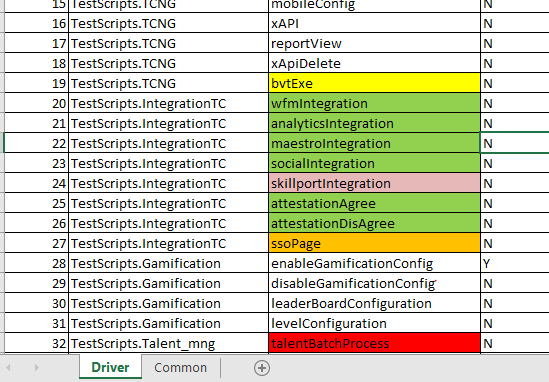
* Install Eclipse for 64 bit IDE
* Create project workspace and launch eclipse IDE
* Import project by right click on package explorer panel and click on import link.
* Select existing projects into workspace
* Browse and select the workspace and project folder.
* Select the project and click on finish button.
* Configure build path and libraries and external Jars by adding below libraries build path.
* JRE system Library
* TestNG library

**External Jar file:**

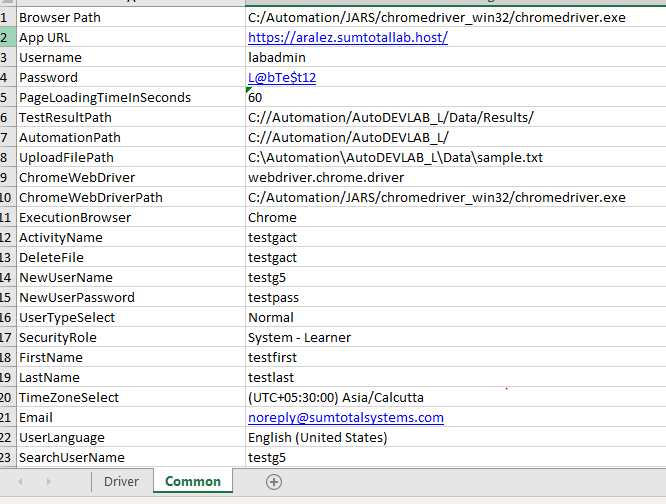
* Selenium Jars
* Extent Report Jars
* POI jars
* Chromedriver\_win32.exe

Execution Steps:

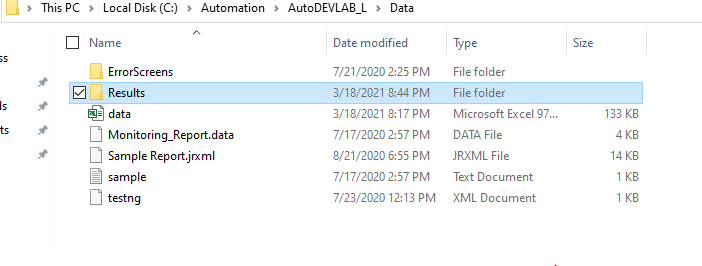
* Open data.xls which is used for Data Driven testing in our framework from the path <project\_name>/Data/data.xls which has Driver and Common sheets.

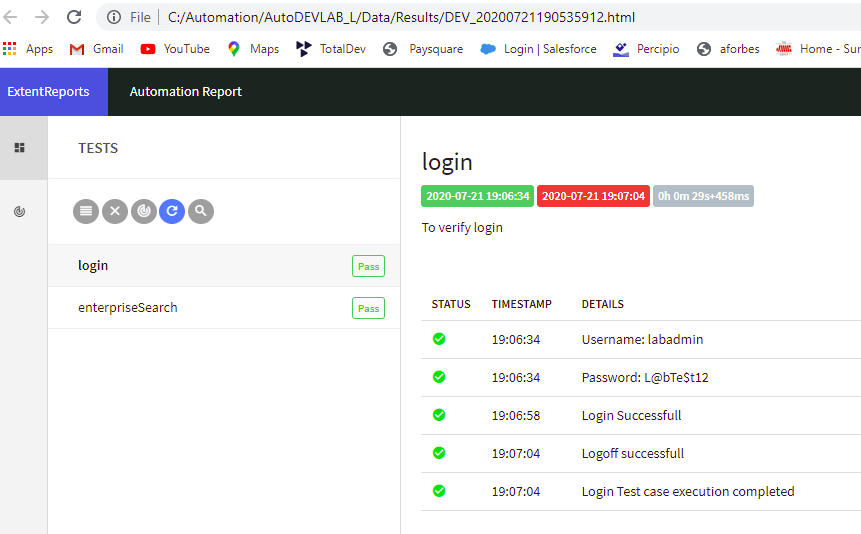


* Common tab stores all user data constraints like application URL, Username, Password, and user inputs required during test script execution.

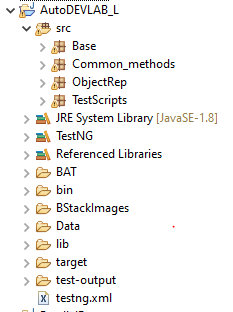


* Driver sheets is the one holds test script names to be executed and Execution ready field to mark the selected scripts to be run during the execution.
* After execution of prescribed test cases the results can be seen in Results folder present in project folder. Screenshots will be captured only during failure ans stored at the results with respective test script name.





Code Structure:



Base package:

Execution starts from Initiator class in Base package.

Common\_methods:

Data extraction from excel and processing happens from this package.

ObjectRep:

As per Page Factory concept, objects are captured at page level and respective methods are coded inside this. Every single page is coded as class and consists of both one time resuable methods. Any change in object description can be modified at one place one time.

TestScripts:

Test cases are grouped by functionality level. Every single method in a class denotes functional test flow. All the test methods are identified by ‘@Test’ TestNG annotation placed on top of it.Each test class has pre-execution setup and post execution setups with ‘@BeforeMethod’ ‘@BeforeTest’, ‘@AfterMethod’, and @AfterTest.